(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平7－262044

(43)公開日　平成７年(1995)10月13日

| (51)Int.Cl.[6] | 識別記号 | 庁内整理番号 | ＦＩ | 技術表示箇所 |
|---|---|---|---|---|
| Ｇ０６Ｆ　11/28 | | Ａ　7313－5Ｂ | | |
| 9/44 | ５３０ | Ｐ　7737－5Ｂ | | |

審査請求　未請求　請求項の数10　ＯＬ　（全　８　頁）

(21)出願番号　特願平6－55418

(22)出願日　平成６年(1994)３月25日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　高舘　公人
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　イベントトレース解析方法

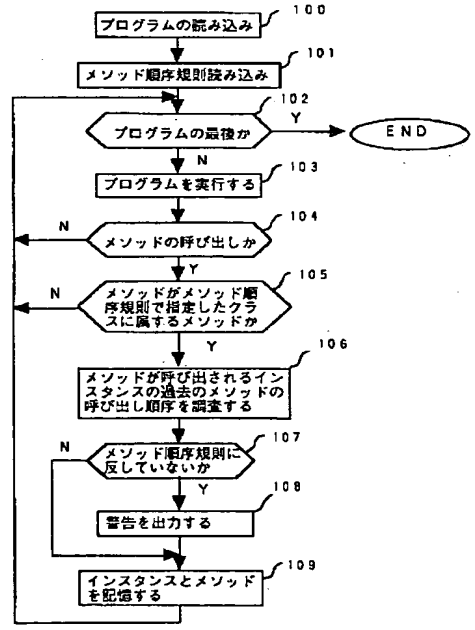
(57)【要約】

【目的】　あるインスタンスに対して誤った順序でメソッドが呼び出されていることをチェックする。

【構成】　検査するプログラムを読み込み(100)、利用者が定義したメソッドの呼び出し順序規則情報を読み込む(101)。プログラムの最後まで(102)、１文ずつ実行し(103)、実行した文が、メソッドの呼び出し文ならば(104)、メソッドが呼び出し順序規則で指定したクラスに属するか否かを調査し(105)、属するならばインスタンスに対するメソッドの呼び出し順序を調査する(106)。もし順序規則に反した呼び出しを行っていれば(107)、警告を利用者に表示する(108)。

【効果】　プログラムのバグを早期に発見する。

図1



1

【特許請求の範囲】

【請求項１】プログラム内の関数の呼び出し関係を解析し表示するシステムにおいて、利用者が定義した、あるクラスのメソッドの呼び出し順序に関するルールを有し、プログラムを実行して、メソッドの呼び出し順序を記憶し、該ルールに反する条件が成立した際に、利用者にメッセージを出力することを特徴とするイベントトレース解析方法。

【請求項２】利用者が定義した、あるクラスの属性の値の遷移順序に関するルールを有し、プログラムを実行して、前記属性の値の遷移順序を記憶し、該ルールに反する条件が成立した際に、利用者にメッセージを出力することを特徴とする請求項１記載のイベントトレース解析方法。

【請求項３】メソッドの呼び出し順序、属性の値の遷移順序が正規表現を使用して表現することを特徴とする請求項１、２記載のイベントトレース解析方法。

【請求項４】メソッドの呼び出し順序、属性の値の遷移順序が状態遷移図を使用して表現することを特徴とする請求項１、２記載のイベントトレース解析方法。

【請求項５】呼び出し元のメソッドＡが参照／更新する属性を記憶し、メソッドＡが直接あるいは間接に呼び出しているメソッドの中で更新している属性を検索し、検索した該属性が、メソッドＡが参照更新している属性と一致した時に、メッセージを表示することを特徴とする請求項１記載のイベントトレース解析方法。

【請求項６】クラスの分類規則と呼び出しを許可しているクラスの集合間の関係を有し、プログラムを解析した結果、呼び出しを許可していないクラス間のメソッドの呼び出しが存在した際に、メッセージを出力すること特徴とする請求項１記載のイベントトレース解析方法。

【請求項７】プログラムの実行中に、生成されたインスタンスと消滅したインスタンスを記録し、あるメソッドが既に消滅されたインスタンスにメッセージを送信した際に、利用者にメッセージを出力すること特徴とする請求項１記載のイベントトレース解析方法。

【請求項８】プログラムの実行中に、メソッドが呼び出される度に、クラス名とメソッド名を表示することを特徴とする請求項１記載のイベントトレース解析方法。

【請求項９】利用者がモニタしたいクラスの名称あるいはクラスのグループの名称を入力し、プログラムの実行中に、利用者が指定したクラスのメソッドが呼び出される度に、メソッド名を表示することを特徴とする請求項８記載のイベントトレース解析方法。

【請求項１０】プログラムの実行中に、メソッドが呼び出される度に、クラス名とメソッド名を表示する際に、さらに利用者の定義したルールと照らし合わせ、ルールに反する条件が成立した際に、メソッド名を明示して表示すること特徴とする請求項８記載のイベントトレース解析方法。

2

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、プログラムのデバッグを支援するシステムに関する。特に、データとデータにアクセスする関数を１つのクラスと見なし、クラスの集合としてプログラムを組織化する、オブジェクト指向プログラムのデバッグに有効なシステムに関する。オブジェクト指向では、データを属性と呼び、関数をメソッドと呼ぶ。

【０００２】

【従来の技術】特願昭６２－０７６０７９に見られるように、プログラムをデバッグする際に、関数の呼び出し順序を利用者が定義し、利用者が定義した順序とは異なる順序で関数が呼び出された時に、警告を出すものがあった。例えば、利用者が定義した関数の呼出し順序が、ＭＡＩＮ→ＳＵＢ１→ＳＵＢ２→ＳＵＢ３→ＳＵＢ４であった時に、ＳＵＢ２の次にＳＵＢ４が呼び出されると、利用者に警告を表示する。

【０００３】

【発明が解決しようとする課題】オブジェクト指向のプログラムでは、クラスがプログラムを組織する単位になっており、プログラムの実行中にクラスからインスタンスが生成され、インスタンス間でメソッドを呼びだすことで処理が行われる。各メソッドはクラス内の属性の値を参照したり更新したりする。各クラスの中のいくつかの属性の値が状態を形成し、クラスは固有の状態遷移を有する。クラスは、状態遷移を引き起こすいくつかのメソッドを有し、これらのメソッドが呼ばれる順序は、クラスの状態遷移の仕様に依存し、誤った順序で呼ばれるとプログラムの誤りとなる。例えば、図書館で管理する本を扱う、クラス本を考える。クラス本には、貸し出し可能、貸し出し中、廃棄の３つの状態がある。クラス本に属するインスタンスは、図書館で管理する本１つ１つであり、それぞれ、貸し出されたり、返却されることで、状態が変化する。プログラムにおいては、各インスタンスは、メソッドが呼びだされることで、個々に状態遷移する。

【０００４】従来の技術は、関数の呼び出し順序に着目し、順序を予め記述していた。しかし、１つのインスタンスに着目してメソッドの呼び出し順序を記述することができなかった。例えば、クラスＢｏｏｋはメソッドＬｅｎｄ、メソッドＲｅｔｕｒｎ、メソッドＤｅｌｅｔｅを有するとする。Ｌｅｎｄ、Ｒｅｔｕｒｎの順に呼び出さなければならいというルールが存在するとする。クラスＢｏｏｋは様々なインスタンスを有し、それぞれに状態遷移がことなる。メソッドＬｅｎｄが続けて呼び出されても、それが別のインスタンスに対するのメソッド呼び出しならば、誤りとはならない。従来技術では、このような状況を区別できず、誤りと判定していた。

【０００５】また、あるクラスＡのメソッドａから直接

あるいは間接的に呼びだされるクラスAのメソッドの中に、メソッドaが参照あるいは更新しているクラスAの属性を、更新しているメソッドが存在しているとすると、バグの原因になりやすい。従来のデバッガでこのような状況を発見するのは困難であった。

【０００６】

【課題を解決するための手段】オブジェクト指向のプログラムは、複数のクラスの集合として、プログラムが構築される。開発者はクラス単位に設計、実装、テストを進めていく。従って、クラスの単位で条件を記述するのは考えやすい。様々なパターンのメソッドの呼び出し順序も、１つのクラスの中のメソッドの呼び出し順序に限定して記述すれば容易である。

【０００７】第１の発明は、オブジェクト指向のプログラムのためのデバッグ支援方法であり、利用者が定義した、１つのクラスのメソッドの呼び出し順序に関するルールを読み込んで、プログラムを実行する際に、利用者が指定したクラスのメソッドの呼び出し順序を記憶していく。利用者が指定した呼び出し順序と反した時に、メッセージを出力する。ルールはメソッドの正規表現による順序あるいは、状態遷移図によって表現される。これによって、オブジェクト指向プログラムにおける誤ったメソッドの呼び出しが発見できる。

【０００８】さらに、メソッドだけでなく、クラスが有する属性の値の遷移に関するルールを読み込んで、これに反した時に、メッセージを出力する。これによっても、誤ったメソッドの呼び出しが発見できる。

【０００９】第２の発明は、属性とメソッドの関係に着目し、あるメソッドａで参照あるいは更新しているクラスの属性を記憶し、あるメソッドａから直接あるいは間接的にメソッドが呼ばれる際に、呼び出されたメソッドの更新する属性を抽出し、メソッドａが参照あるいは更新している属性が含まれていないかチェックする。もし、含まれていれば、メッセージを表示する。これによって、メソッドの設計ミスが発見できる。

【００１０】

【作用】第１の発明では、利用者はメソッドの呼び出し順序を正規表現、あるいは状態遷移図で表現する。例えば、クラスAにおける正規表現でａ１｛ａ２｝ａ３を、初めにａ１が呼ばれて、次にａ２が１回以上呼ばれて、最後にａ３が呼ばれる意味とする。利用者が定義したメソッドの呼び出し順序を読み込み、プログラムの実行中に、メソッドの呼び出しを監視する。クラスAのインスタンスであるａに対して、ａ１がまず呼ばれると、インスタンスａに対して、ａ１が呼ばれたことを記憶する。他のクラスのメソッド、あるいはクラスAのａ１、ａ２、ａ３以外のメソッドの呼び出しは無視する。次に、インスタンスａに対して、ａ２が呼ばれるとａ２が呼ばれたことを記憶する。次に、ａ１が呼ばれると正規表現と異なるので、メッセージを出力する。これによって、



プログラムが誤っていることが判る。

【００１１】第２の発明では、あるメソッドａで参照あるいは更新している属性を記憶し、該メソッドから直接あるいは間接に呼び出しているメソッドｂの更新している属性と比較するので、メソッドａで使用している属性を、更新してしまっているメソッドが分かる。開発者が意図的に行っている場合は問題無いが、意図に反している場合は、バグの原因となる。バグの原因になりそうな事象を、利用者に示すことにより、プログラムのデバッグを支援する。

【００１２】

【実施例】以下、本発明の実施例を詳細に説明する。最初に第１の発明の実施例を説明する。第１の発明は、１つのクラスに対するメソッドの呼び出し順序が、利用者の意図と反した際に、すぐに発見できるため、プログラムのバグが見つけやすくなる。

【００１３】第１の発明を実現するシステムのフローチャートを図１に示す。図２は本発明のハードウェア構成図を示す。２００はプログラムやメソッドの呼び出し順序に関する規則を入力する入力装置、２０１は本発明で解析するプログラムとデータを一時的に記憶する主記憶装置、２０２は本発明のプログラムを実行するＣＰＵ、２０３は本発明のプログラムや解析対象となるプログラムやデータを記憶する外部記憶装置、２０４は本発明の出力結果を表示するＣＲＴである。

【００１４】図１のフローチャートに基づいて本実施例を説明する。検査するプログラムを読み込み(100)、利用者が定義したメソッドの呼び出し順序規則情報を読み込む(101)。図書館で管理する本のクラスにおけるメソッドの呼び出し順序規則を正規表現で記述した例を以下に示す。

【００１５】Ｂｏｏｋ：：｛Ｌｅｎｄ，Ｒｅｔｕｒｎ｝＋，Ｄｅｌｅｔｅ

最初にクラス名が示され、’：：’で区切られた後に、メソッドの呼び出し順序が示される。’｛｝’は組合せを表し、’＋’は１回以上を表す。ＬｅｎｄとＲｅｔｕｒｎの組合せが１回以上呼び出された後に、最後にＤｅｌｅｔｅが呼び出されて終了する。つまり、本の貸出しと返却が繰り返されて、最後に本が破棄されることを意味している。このような呼び出し関係が定義された際に、Ｌｅｎｄが２度続けて呼び出されたり、Ｒｅｔｕｒｎが２度続けて呼び出されると、プログラムのエラーとなる。順序規則情報を読み込んだ後に、プログラムの最後まで(102)、１文ずつ実行する(103)。実行したプログラムの文が、メソッドの呼び出し文ならば(104)、メソッドが、呼び出し順序規則で指定したクラスに属するか否かを調査する(105)。クラスＢｏｏｋのメソッドＬｅｎｄが、インスタンスｂｏｏｋ２に対して呼び出されたとすれば、Ｌｅｎｄは順序規則で指定したクラスに属するメソッドとなる。呼び出しメソッドテーブルを調査

し、インスタンスに対するメソッドの呼び出し順序を調査する(106)。もし順序規則で指定したクラスに属するインスタンスで、順序規則に反した呼び出しを行っていれば(107)、警告を利用者に表示する(108)。表１に呼び出しメソッドテーブルを示す。本テーブルは、現在までに呼び出されたメソッド名とクラス名、呼び出しの対象となったインスタンス名を示す。

【００１６】

【表１】

表１

呼び出しメソッドテーブル

| インスタンス名 | クラス名 | メソッド名 |
|---|---|---|
| ｂｏｏｋ１ | Ｂｏｏｋ | Ｌｅｎｄ |
| ｂｏｏｋ２ | Ｂｏｏｋ | Ｌｅｎｄ |
| ｕｓｅｒ１ | Ｕｓｅｒ | Ｌｅｎｄ |
| ｂｏｏｋ１ | Ｂｏｏｋ | Ｒｅｔｕｒｎ |

【００１７】ｂｏｏｋ２に対しては、過去にＬｅｎｄが呼び出されており、再びＬｅｎｄが呼び出されたので、メソッドの順序規則に反する。よって、警告を出力する。その後、呼び出したメソッドを記憶し(108)、102に戻ってプログラムを実行する。

【００１８】呼び出し順序は、状態遷移図でも表現できる。状態遷移図で記述した呼び出し順序の例を図３に示す。意味するところは、上記の正規表現と同じであり、本は、貸出し可能と、貸出し中の２つの状態を交互にとり、２つの状態を変化させるメソッドがＬｅｎｄとＲｅｔｕｒｎであることを示している。そして、最後にＤｅｌｅｔｅのメソッドが呼ばれて、廃棄の状態になることを示している。

【００１９】次に第２の発明の実施例を図４のフローチャートを用いて説明する。第２の発明は、あるメソッドから直接的あるいは間接的に呼び出されているメソッドが、同じ変数を、同時に更新している時に発生する、プログラムのバグを早期に発見できる。

【００２０】プログラムを読み込み、実行して、実行したプログラム文がメソッドの呼び出しならば(104)、呼び出したメソッドが更新している属性を検索する(400)。例えば、インスタンスｕｓｅｒ１に対して、ＵｓｅｒクラスのメソッドＢｏｒｒｏｗが呼び出されて実行された時に、メソッドＢｏｒｒｏｗで更新している属性がｃであったとする。次に参照更新テーブルの中から、インスタンス名、クラス名が同一であり、かつ現在呼び出したメソッドの中で更新している属性を参照あるいは更新しているメソッドを検索する(401)。参照更新属性テーブルの例を図５に示す。５００はインスタンス名を記憶する列、５０１はクラス名を記憶する列、５０２はメソッド名を記憶する列、５０３はメソッドの中で参照している属性を記憶する列、５０４はメソッドの中で更新



している属性を記憶する列を示す。メソッドが呼び出された順に上から記憶される。ＵｓｅｒクラスのメソッドＢｏｒｒｏｗはｃを更新しているので、Ｕｓｅｒクラスであり、かつ属性ｃを参照あるいは更新しているメソッドを検索する。図５では、過去にｕｓｅｒ１に対して、ＵｓｅｒクラスのＬｅｎｄが呼ばれており、Ｌｅｎｄの中で更新されている属性は、ｃであるのでＬｅｎｄのメソッドがこれに該当する。よって、警告を出力する(403)。次に、呼び出されたメソッドが参照更新している属性を、参照更新属性テーブルに記憶し(404)、再び102から繰り返す。実行したプログラムの文が、メソッドの終了文ならば(405)、終了したメソッドに対応する参照更新属性テーブル内のレコードを削除する(406)。

【００２１】次に、第３の発明の実施例を図６のフローチャートを用いて説明する。第３の発明は、呼び出しを禁止しているクラスの間でメソッドの呼び出し関係があるかないかをチェックする。例えば、様々なアプリケーションに共通的に利用できるクラスと、あるアプリケーションに固有なクラスが存在したときに、固有なクラスから共通利用できるクラスを呼びだすことはできるが、共通利用できるクラスから固有なクラスは呼び出しができない。なぜなら、共通利用できるクラスから固有なクラスを呼び出してしまうと、呼び出しの部分があるアプリケーションに固有な部分になってしまい、クラス全体が共通利用できなくなってしまうためである。本発明はこのような状況を防ぐためのものである。

【００２２】プログラムを読み込み、利用者が定義した呼び出し関係テーブルを読み込む(600)。呼び出し関係テーブルの例を図７に示す。７００は呼び出し元のクラスあるいはクラスのグループを示し、７０１は呼び出し元のクラスから呼び出し可能なクラスのグループあるいはクラスを示す。図７の最初のレコードは、クラス名の先頭が”ＡｗａＳ”になっているクラスから、クラス名の先頭が”ＡｗａＳ”になっているクラスを呼び出すことが可能であることを示している。

【００２３】実行したプログラム文がメソッドの呼び出しならば(104)、呼び出し関係テーブルから呼び出し関係を取得し(601)、呼び出し元のメソッドが呼び出し関係テーブルに記述されたクラスあるいはクラスグループに含まれているかチェックする(602)。もし含まれていれば、呼び出されたメソッドが呼び出しを許すクラスに含まれているかチェックする(604)。例えば、クラスＡｗａＳＢｏｏｋのメソッドＬｅｎｄから、クラスＡｗｅＳＵｓｅｒのＬｅｎｄが呼ばれたとする。先頭にＡｗａＳが付くクラスから呼び出し可能なクラスは、先頭にＡｗａＳあるいはＡｗａＵが付くクラスである。ＡｗｅＳＵｓｅｒのメソッドは呼び出し可能なクラスのメソッドではない。もし呼び出しが不可能なクラスのメソッドを呼び出している時は、警告を出力する(605)。

【００２４】次に、第４の発明の実施例を図８のフロー

チャートを用いて説明する。オブジェクト指向のプログラムでは、インスタンス間でメソッドを呼び出しながら、処理を行う。その際に、既に消滅してしまったインスタンスに対して、メソッドを呼び出すとプログラムが暴走する。第４の発明は、既に消滅してしまったインスタンスに対して、メソッドを送ることによるプログラムの誤りを早期に発見する。　実行したプログラム文がメソッドの呼び出しならば(104)、呼び出し先のインスタンスが存在するか否かをインスタンステーブルでチェックする(800)。表２にインスタンステーブルの例を示す。

【００２５】

【表２】

表２

呼び出しメソッドテーブル

| インスタンス名 |
|---|
| ｂｏｏｋ１ |
| ｂｏｏｋ２ |
| ｕｓｅｒ１ |

【００２６】インスタンステーブルは存在しているインスタンスを管理する。もし、存在しないインスタンスにメソッドを呼び出していたら(801)、警告を表示する(802)。実行したプログラム文がインスタンスを消滅させる文ならば(803)、インスタンステーブルからインスタンスを削除する(804)。インスタンスを生成させる文ならば(805)、生成したインスタンスをインスタンステーブルに追加する(806)。

【００２７】次に、第５の発明の実施例を図９のフローチャートを用いて説明する。プログラムに誤りがあった際に、メソッドの呼び出し順序を調べることは、バグの発見に役に立つ。第５の発明は、バグが含まれるプログラムを実行することで、クラス名とメソッド名を呼び出された順に表示するので、バグの発見に役に立つ。

【００２８】プログラムを読み込み、実行して、実行したプログラム文がメソッドの呼び出しならば(104)、クラス名とメソッド名を表示する(900)。表示した結果を図１０に示す。ＢＥＧＩＮはメソッドが呼び出されたことを示し、ＥＮＤは呼び出されたメソッドの実行が終了したことを示す。インデンテーションは、あるメソッドから別なメソッドが呼び出されていることを示す。例えば、クラスＡｗａＳＵｓｅｒのＬｅｎｄのメソッドから、クラスＡｗｅＳＢｏｏｋのＬｅｎｄのメソッドが呼び出されていることを示している。

【００２９】第６の発明は、第５の発明の応用である。第５の発明は、全てのメソッドの呼び出しについて出力されるが、大きなプログラムでは、メソッドの呼び出し回数は非常に多く、出力結果が膨大になり、出力結果を表示するのに時間を要する。利用者がクラスあるいはクラスのグループを指定することにより、指定したクラスに属するメソッドのみを表示することができ、誤りの発見がより早くなる。



【００３０】プログラムと、メソッド名を表示するクラスのグループ名を読み込む(1100)。実行したプログラム文がメソッドの呼び出し文であるときに(104)、利用者が指定したクラスグループに属するクラスのメソッドか否かを調査し(1101)、もし属すればクラス名、メソッド名を表示する(900)。例えば、利用者がクラスのグループとして先頭がＡｗａＳで始まるクラスを指定したとすると、図１０において、ＡｗａＳＵｓｅｒの呼び出しと、ＡｗａＳＴａｂｌｅの呼び出しのみが表示される。

【００３１】第７の発明は、第５の発明と、第１、第２、第３、第４のそれぞれの発明との組合せである。第１、第２、第３、第４の発明において、誤ったメソッドの呼び出しの際に警告が出されるが、警告を第５の発明におけるメソッドの呼び出しの出力の上で行うものである。例えば、誤ったクラス間の呼び出しを表示する第３の発明と組み合わせる。図１０の例では、共通利用できるクラスであるＡｗａＵｓｅｒから、アプリケーションに固有なクラスであるＡｗｅＳＢｏｏｋのメソッドが呼ばれているので、誤りである。よって、図１２では誤ったメソッドの呼び出し部分を強調表示している。

【００３２】

【発明の効果】第１の発明では、プログラムを実行し、あるインスタンスに対して、利用者が定義したメソッドの呼び出し順序と異なった順序でメソッドが呼び出されていることが判る。これによって、プログラムが誤っていることが判り、早期にバグの原因が発見できる。

【００３３】第２の発明では、あるメソッドａで参照あるいは更新している属性を記憶し、該メソッドから直接あるいは間接に呼び出しているメソッドｂの更新している属性と比較するので、メソッドａで使用している属性を、更新してしまっているメソッドが分かる。開発者が意図的に行っている場合は問題無いが、意図に反している場合は、バグの原因となる。バグの原因になりそうな事象を、利用者に示すことにより、プログラムのデバッグを支援する。

【図面の簡単な説明】

【図１】本発明の第１の実施例のフローチャートである。

【図２】本発明のハードウェア構成図である。

【図３】本発明の第１の実施例で用いるメソッド順序規則に関するルールである。

【図４】本発明の第２の実施例のフローチャートである。

【図５】本発明の第２の実施例で用いるメソッドとメソッド内で参照更新する属性の対応表である。

【図６】本発明の第３の実施例のフローチャートである。

9

【図７】本発明の第３の実施例で用いる呼出しを許すメソッドの対応関係表である。

【図８】本発明の第４の実施例のフローチャートである。

【図９】本発明の第５の実施例のフローチャートである。

【図１０】本発明の第５の実施例の出力例である。

【図１１】本発明の第６の実施例のフローチャートである。

【図１２】本発明の第７の実施例の出力例である。

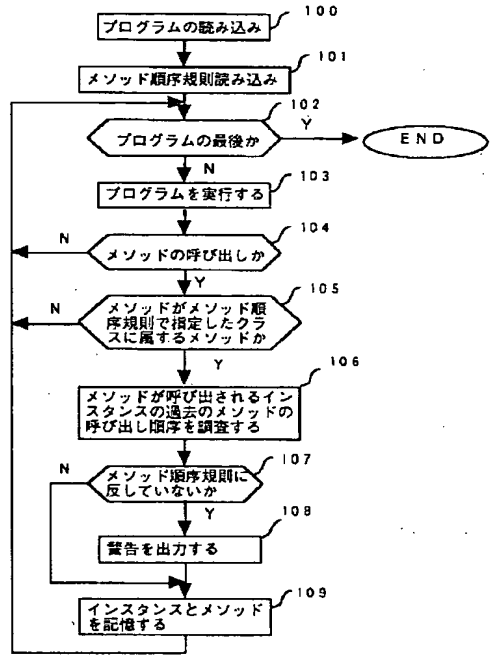
【符号の説明】

10

＊１００：プログラムの読み込み処理、１０１：メソッドの順序規則の読み込み処理、１０２：プログラムの最後を判定する処理、１０３：プログラムの実行処理、メソッドの呼出しを判定する処理、１０５：メソッドが順序規則で規定されたメソッドか否かを判定する処理、１０６：呼び出したメソッドが順序規則に反していないかチェックする処理、１０７：メソッドが呼び出されるインスタンスの過去のメソッドの呼出し順序を調査する処理、１０８：規則に反していた際に警告を出力する処理、１０９：呼び出したメソッドを記憶する処理。

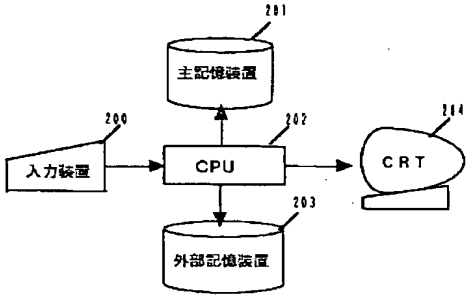
【図１】

図1



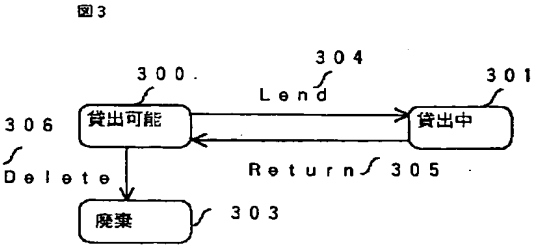

【図２】

図2



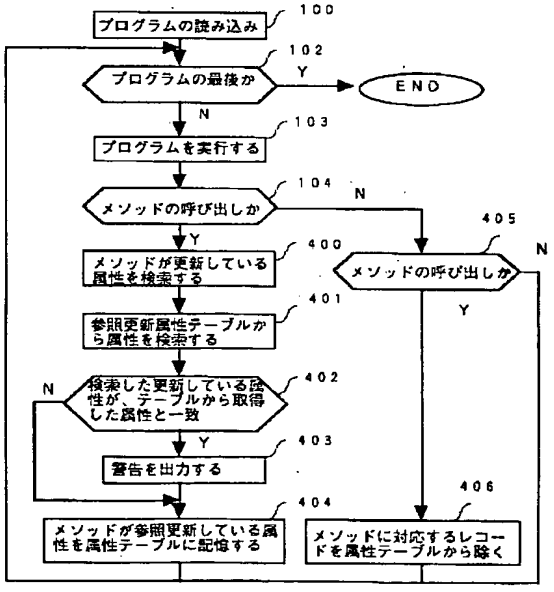

【図３】

図3




【図４】

図4



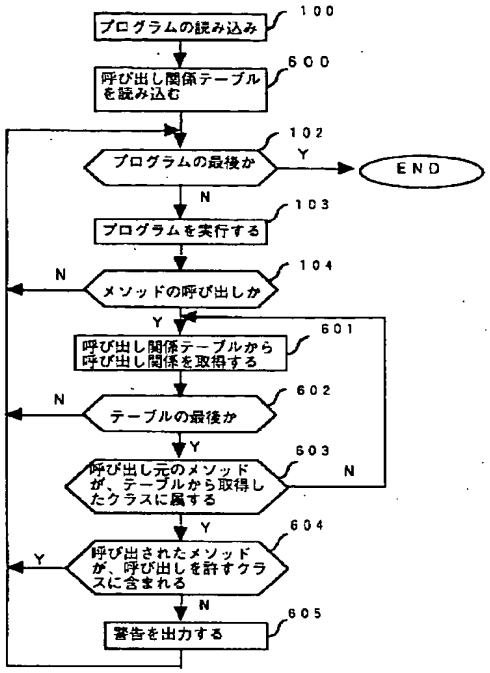
【図５】

図5

| 500 インスタンス名 | 501 クラス名 | 502 メソッド名 | 503 参照属性名 | 504 更新属性名 |
|---|---|---|---|---|
| user1 | User | Lend | a、b | c |
| book1 | Book | Lend | d | e |
| table1 | Table | Entry | f、g | h |

【図６】

図6



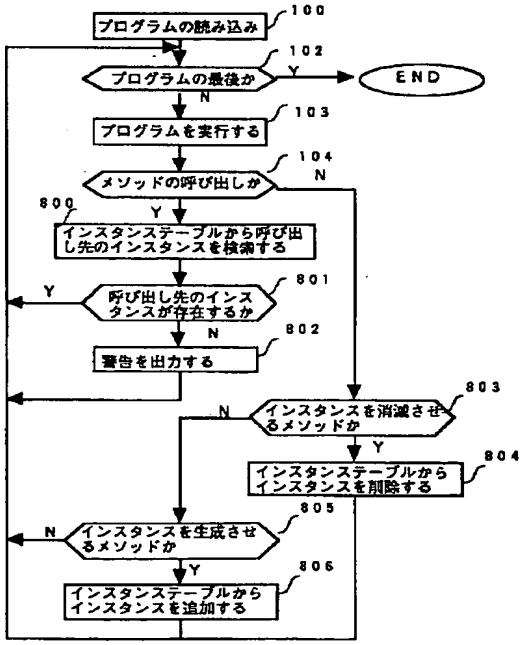

【図７】

図7

| 700 呼び出し元クラス | 701 呼び出し先クラス |
|---|---|
| AwaS* | AwaS* |
| AwaU* | AwaU* |
| AweS* | AwaS* |
| AweS* | AweS* |
| AweU* | AwaU* |
| AweU* | AweU* |

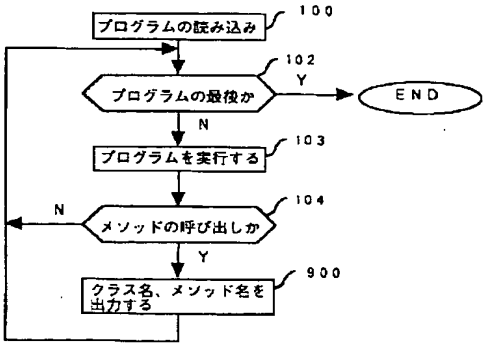
【図８】

図8




【図９】

図9



【図１０】

図10

```
+BEGIN  AwaSUser::Lend
|  +BEGIN  AweSBook::Lend
|  -END    AweSBook::Lend
|  +BEGIN  AwaSTable::Entry
|  -END    AwaSTable::Entry
-END    AwaSUser::Lend
+BEGIN  AwaSUser::Return
|  +BEGIN  AweSBook::Return
|  -END    AweSBook::Return
|  +BEGIN  AwaSTable::Delete
|  -END    AwaSTable::Delete
-END    AwaSUser::Return
```

【図１２】

図12

```
+BEGIN  AwaSUser::Lend
|  +BEGIN  AweSBook::Lend
|  -END    AweSBook::Lend
|  +BEGIN  AwaSTable::Entry
|  -END    AwaSTable::Entry
-END    AwaSUser::Lend
+BEGIN  AwaSUser::Return
|  +BEGIN  AweSBook::Lend
|  -END    AweSBook::Lend
|  +BEGIN  AwaSTable::Delete
|  -END    AwaSTable::Delete
-END    AwaSUser::Return
```

【図１１】

図11